(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
11 December 2003 (11.12.2003)

PCT

(10) International Publication Number
WO 03/102806 A1

(51) International Patent Classification[7]: G06F 17/00, H04L 29/00

(21) International Application Number: PCT/IB02/02936

(22) International Filing Date: 31 May 2002 (31.05.2002)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant *(for all designated States except US)*: **NOKIA CORPORATION** [FI/FI]; Keilalahdentie 4, FIN-02150 ESPOO (FI).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **PESSI, Pekka** [FI/FI]; Neitojenranta 1 A 9, FIN-00810 Helsinki (FI).

(74) Agents: **WILLIAMS, David, John** et al.; Page White & Farrer, 54 Doughty Street, London WC1N 2LS (GB).

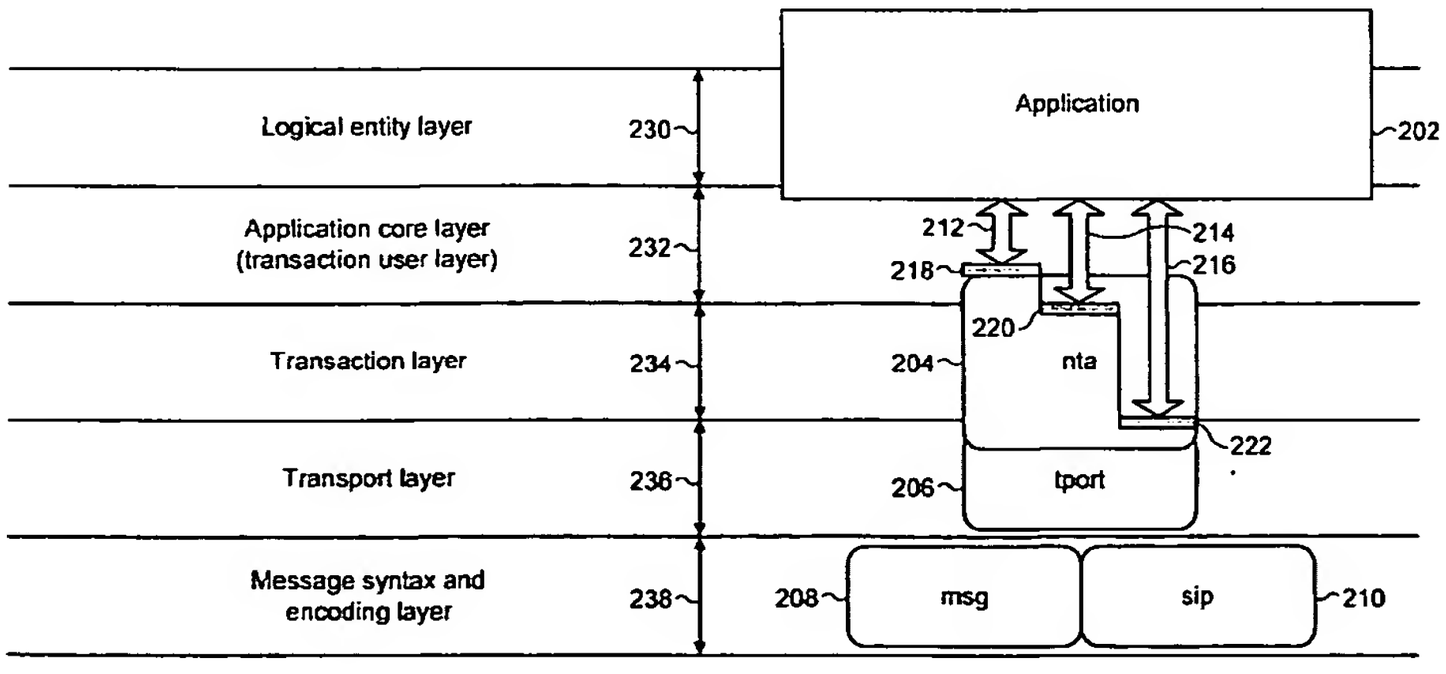
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: PROTOCOL ENGINE APPLICATION INTERFACE



(57) Abstract: There is disclosed an interface block for providing application access to the underlying protocol engine at several different layers. These layers are preferably the application core layer, the transaction layer, and the transport layer.

WO 03/102806 A1

PROTOCOL ENGINE APPLICATION INTERFACE

Field of the Invention

The present invention relates to applications which use underlying protocol engines. The invention is particularly but not exclusively related to multimedia applications, and SIP, the IP telephony signaling protocol.

Background to the Invention

Multimedia applications are increasingly utilised, particularly in mobile wireless applications.

In any such applications, an application must have access to an underlying protocol engine. A common protocol engine in IP (Internet protocol) telephony is the Session Initiation Protocol (SIP).

The most popular API specification for SIP is known as JAIN SIP API. It is a collection of Java interfaces implementing access to SIP protocol engines at a transaction level.

The JAIN SIP API is inflexible, hard to extend, underspecified and does not suit well proxy applications.

It is therefore an object of the present invention to provide an improved access for application to an underlying protocol engine, particularly but not exclusively for the case where that protocol engine is a SIP engine.

Summary of the Invention

In accordance with the present invention there is provided a method of providing access from an application to an underlying protocol engine at a plurality of different layers.

Said access is preferably provided through a transaction layer.

Access may be provided through an interface of the transaction layer at a plurality of different layers.

Access may be provided at an application core layer. Said access may be to calls.

Access may be provided at a transaction layer. Access may be to transactions.

Access may be provided at a transport layer. Said access may be to a stateless message transport.

There may further additionally be provided other application core functionality.

There may further be provided session initiation protocol functionality to the application.

The application may be IP telephony.

A request message may be processed, as is well known, in dependence on a comparison of the top most Via header of the message. At least part of the request routing may implemented as part of a dialog matching code. There may be provided a failure-proof fallback code for routing and processing logic.

The application may be a multimedia application.

In accordance with the present invention there is further provided an interface adapted to provide access between an application and an underlying protocol engine at a plurality of different layers.

The interface may be formed in a transaction layer. The transaction layer may be adapted to provide an interface at the plurality of different layers.

An interface may be provided at an application core layer. The interface may be provided at a transaction layer. The interface may be provided at a transport layer.

There may further be provided means for implementing further application core functionality above the interface.

There may further be provided means below the interface functionality.

The invention is thus a software interface and underlying middleware for SIP. It provides an easy-to-use and flexible framework for creating SIP applications. Its features make it easy to create correct and robust SIP software that can make use of later protocol extensions.

The invention provides versatile means for a IP telephony application writer to add SIP functionality to their application. While the applications may be very different from each other, it is beneficial to use common middleware components implementing the core SIP protocol functionality.

The interface provides versatile access to an SIP protocol engine in particular. The application (e.g. proxy server, redirect server) can be stateful or stateless, the application can itself assemble all messages or it can let the interface complete them. The application can let the interface filter out extra messages not affecting the application state, like 100 Trying. Most function calls take extra named arguments, thus making it easy to pass extra arguments to lower layers taking care of transport or message syntax.

The applications can customize the parsing process. For example, they can avoid parsing irrelevant headers or cache the message contents if it will be forwarded. Adding new headers is trivial, and does not affect the interface.

Because implementation allows the application to create new header classes at run time, it is possible to add new headers without compiling anything.

Applications can let the interface take care of tasks belonging to the application core layer, like routing requests or responses according to DNS data, matching incoming requests with dialogs, or matching incoming requests with existing server transaction objects.

Brief Description of the Drawings

The invention is now described by way of example with reference to the accompanying figures in which:

Figure 1 shows in block diagram form an example architecture of a multimedia application;

Figure 2 illustrates the logical layers and their interconnectivity in an exemplary embodiment of the present invention;

Figure 3 illustrates a further modification to the exemplary embodiment of Figure 2;

Figure 4 illustrates a comparison of the advantages of the present invention over the prior art; and

Figure 5 illustrates a state diagram associated with an exemplary embodiment of the invention.

Description of the Preferred Embodiments

The present invention is described herein by way of reference to a particular non-limiting example. One skilled in the art will appreciate that the invention may be more broadly applicable.

The invention is described herein with particular reference to an implementation in a software for internet applications

(SOFIA) implementation. A block diagram of an exemplary SOFIA architecture is shown in Figure 1, including a signalling subsystem adapted in accordance with a preferred embodiment of the present invention.

It is assumed in the following discussion that the skilled reader is familiar with the well-known session initiation protocol (SIP), session description protocol (SDP) and other signalling protocols as discussed hereinbelow. The implementation of such protocols does not form part of the present invention in so far as such implementations are not described herein.

Referring to Figure 1, the SOFIA architecture includes a set of applications 102, a signalling subsystem 104, a media subsystem 106, and an operating system abstraction layer 108.

The applications 102 include, in the example shown, a proxy server application 110, a registrar/presence server application 112, a simple ubiquitous rich-call facilitator (SURF) server application 114, and media server applications such as announcement server 116 and conference server application 118.

The signalling subsystem 104 includes iptres, iptsec, and nea blocks 119, 121 and 123 respectively, a Nokia user agent API (NUA) 122, a Nokia Transaction API (NTA) 130, a nth 124, a ntr 126, a transport block 127, an IP telephony utility library (IPT) 138, a http block 132, a SIP block 134, a RTSP block 136, and a protocol independent message block (MSG) 140. The nea block 123 is a Nokia event API block. The NTA module 130 implements the SIP dialogs and transactions. The tport module 106 implements message transport. The sip module 110 provides syntax and encoding for different SIP

headers. The msg module 108 provides generic SMTP-like abstract syntax and encoding primitives. The tport 106 and msg 108 modules can be shared by other protocols, like RTSP or HTTP.

The elements of the media subsystem 106 include a multimedia subsystem block 142, which realises the interface with the signalling subsystem 104, an SDP block 144 for defining session, media and codec descriptions, an RTP block 148 for transporting media over IP, and including a jitter buffer, packet video module 146 for video conferencing and an audio module 152. The audio module is further associated with an audio device 135, a codec 131, and an RTP block 133. The video module 146 is further associated with a codec 137, a video device 141, and an RTP block 139.

The operating system abstraction layer 108 includes an SU block 158 containing an SU library.

In Figure 1, the signalling subsystem 104 may be considered to be a control part of the application, and the media subsystem may be considered to be a media part of the application.

A detailed description of a preferred embodiment of the present invention is given hereinbelow.

The NTA block 130 is an application programming interface (API) between an IPT application and a transaction-layer session initiation protocol (SIP) protocol engine. It should be noted that although the present invention is described by way of reference to a specific implementation which utilises the Nokia Transaction API, the skilled person reading the following description will appreciate that the functionality provided by the present invention may be more broadly applied.

The NTA block 130 provides means, in accordance with a preferred embodiment of the present invention, for an application to:

1. Establish transport protocol endpoint(s), which are used to receive and send SIP messages;
2. Create client transactions;
3. Process and respond to server transactions;
4. Create, send and respond to SIP messages (in stateless mode); and
5. Create dialogs or dialog templates.

Referring to point 1 above, the transport protocol endpoints (also known as sockets) are established when an NTA agent object (nta_agent_t) is created. The application can control what kind of sockets are created by supplying a URL as a parameter to the NTA block 130. According to the URL, the sockets can be bound to well-known SIP port(s), or to ephemeral ports. Also, the URLs specify which transport protocol(s) are supported.

The NTA interface provides access to SIP protocol methods/primitives at three different layers: dialogs at an application core layer, transactions at a transaction layer, and stateless message transport at a transport layer.

With reference to Figure 2, the As shown in Figure 2, there is illustrated an application block 202, a NTA block 204, a transport block 206, a MSG block 208 and a SIP block 210.

As also shown in Figure 2, the SIP protocol is structured to five layers. The lowest layer 238 is the message syntax layer. The next layer 236 is the message transport layer. The next layer 234 is the transaction layer. These three

layers 234,236,238 behave in an identical way in all SIP elements.

On top of transaction layer 234 is the application core layer 232, which defines three different cores: the User Agent Client (UAC) core, the User Agent Server (UAS) core and the proxy core. An application core defines common rules for basic transaction processing and dialog handling, for instance. The SIP protocol specification also defines a few logical entities: User Agent, Registrar, Redirect Server, Stateless Proxy, Stateful Proxy, and Back-to-Back User Agent. A logical entity describes a role in which a SIP application participates to a SIP operation.

Finally, the highest layer is the logical entity layer 230.

In accordance with the preferred embodiment of the present invention, the NTA interface block 204 provides an application access to the underlying protocol engine at three different layers: dialogs at the application core layer 232 as represented by interface 218; transactions at the transaction layer as represented by interface 220; and stateless message transport at the transport layer as represented by interface 222.

A separate piece of middleware (such as the software for Internet package (SOFIA) NUA User-Agent library) may implement the rest of the application core functionality on top of the NTA.

As shown in Figure 2, in this embodiment of the invention the functionality below the NTA API block is provided by four software modules: the nta block 204 itself; the transport bock 206 tport; the SIP block 210 sip; and the MSG block 208 msg.

As explained further hereinbelow, an application can extend the functionality provided by the underlying layers without modifying or affecting the transaction layer.

Different applications use greatly varying numbers of parameters with the NTA API. For example, while the simplest application can just provide the method name and dialog when creating a transaction, a sophisticated application may include many other parameters ranging from Call-Info containing rich caller identification to overriding 100-filtering attributes. The NTA API in accordance with this embodiment of the present invention therefore provides a flexible way to include optional parameters in function call parameters.

The NTA approach preferably uses named parameters. Named parameters are passed using a special macro expanding to a tag/value pair (this pair can be considered as a constructor). For instance, the tag item macro SIPTAG_SUBJECT_STR(sub) expands to tag siptag_subject_str and string sub. The argument lists are typesafe if the definition uses special inline function casting the value to tag_value_t as follows:

```
#define SIPTAG_SUBJECT_STR(v) \
   siptag_subject_str, siptag_subject_str_v(v)
inline tag_value_t siptag_subject_str_v(char const *s) {
  return (tag_value_t)s);
}
extern tag_type_t siptag_subject_str;
```

Outside parameter lists, the tag and the value are stored in an object called tag item, (tagi_t) defined as follows:

```
typedef struct tag_type_s const *tag_type_t;
typedef unsigned long tag_value_t;  /* integral type that
can hold a pointer */

typedef struct {
  tag_type_t  t_tag;
  tag_value_t t_value;
} tagi_t;
```

The tag is a pointer to a special structure specifying the name of the tag and the type of the tag value.

A tag list can refer to another tag list, therefore making it possible to pass the argument list from one layer to another.

There are tags for SIP headers, NTA attributes, URLs, etc. An application can create its own tags to describe, for instance, a new extension header used by it.

For example, initiating an INVITE transaction in an SIP sesion with a given subject and a couple of rich caller identification URIs would look like this:

```
orq = nta_outgoing_tcreate(
        dialog, &process_response, &call_context,
        NULL,
        SIP_METHOD_INVITE,
        NULL, /* RequestURI, From, To, Call-ID, CSeq are
provided by dialog */
        SIPTAG_SUBJECT_STR(subject),
        SIPTAG_CALL_INFO_STR(icon_url),
        SIPTAG_CALL_INFO_STR(calling_card_url),
        SIPTAG_CONTENT_TYPE_STR("application/sdp"),
        SIPTAG_PAYLOAD(sdp),
        NTATAG_100_TRYING(true), /* Don't filter 100 Trying
from next proxy */
        TAG_END());
```

Preferably, the 100 Trying messages of an SIP sesion are filtered by the NTA. In this embodiment, however, the application gives a user complete feedback for the call progress, including the first 100 Trying.

The prototype for nta_outgoing_tcreate() is as follows:

```
nta_outgoing_t *
    nta_outgoing_tcreate(nta_leg_t *leg,
                         nta_response_f *callback,
                         nta_outgoing_magic_t *magic,
                         url_string_t const *route_url,
                         sip_method_t method,
                         char const *method_name,
                         url_string_t const *request_uri,
                         tag_type_t tag, tag_value_t value,
    ...);
```

The su library provides macros (ta_list, ta_start, ta_args, ta_tags, ta_end) that implement the tag lists for different binary architectures.

Some named arguments specify the semantics followed by NTA. The overall semantic attributes are usually specified with nta_agent_set_params() function; in special cases, the semantic attributes can also be provided for each dialogue or transaction separately (like NTATAG_100_TRYING attribute provided in the example above). The semantic attributes include:

- ua: if this attribute is set, INVITE transaction is processed according to UAS/UAC semantics.
- user_via: if this attribute is set, application is responsible for inserting headers.
- pass_100: if this attribute is set, NTA relays also 100 Trying messages to the application. The 100 Trying response messages are not forwarded upstream (towards client)by stateful proxies.
- extra_100: if this attribute is set, NTA generates itself a 100 Trying responses after a retransmitted request is received and 200 ms has elapsed from receiving the original request. A SIP transaction server should respond with a preliminary response within 200ms upon receiving a request. If no other response can be sent, the transaction server can always send a 100 Trying response message.
- timeout_408: if this attribute is set, NTA generates a 408 response message when a client transaction times out.

- merge_482: if this attribute is set, NTA determines if a request has arrived twice to it and when so happens replies with a 483 Request Merged response message.

The application 202 can also provide a message class as a parameter when the NTA agent object is created. The message class is an interface to the SIP message syntax layer. The application can customize the message class according its needs. Also, the transaction or transport modules can further modify the behavior of the message class if, for example, they need to provide extra debugging information.

Referring to Figure 3, there is further illustrated the modification of an SIP application 210 by way of extension 203. An SIP application, in this example, extends the basic SIP parser functionality beyond the basic subset required by the transaction engine.

This is an important aspect of this embodiment of the invention in an environment where either the code footprint or the performance is an issue. A mobile User Agent can reduce the code footprint by only including bare minimum parsing functionality. A high-performance SIP proxy needs only to parse those headers absolutely required by its routing functionality.

Referring to Figure 4, there is illustrated the difference between the prior art and the present invention. Responding to a request using JAIN SIP API (Figure 4(a)) and NTA (Figure 4(b), the NTA takes care of many routine tasks on behalf of the application.

Referring to Figure 4(a), in the prior art in the application layer 402 a message is created, headers copied from the request, a status line and response headers added.

In the present invention, in the application layer 404 a reply is preferably made with status line, response headers.

In the prior art in the UAS core layer 406, it is checked that the message is complete. In the preferred implementation of the present invention, in the UAS core layer 408 the list is prepended with standard headers (e.g. server).

In the prior art in the transaction layer 410, a response is sent, and a retransmission made if required. In the preferred implementation of the present invention in the transaction layer 412, a message is created. Headers are copies from the request, and headers provided by the upper layers added. It is then checked that the message is complete. The response is sent, and retransmitted if required.

The agent object (nta_agent_t), a dialog object or a server transaction object act as message object factories. Applications can obtain message objects already populated by basic headers from them, for instance. An application can also choose not to handle messages at all, but rather provide required information (request method, URI, response code, extra headers) to the NTA and let it create the message, populate it with its own headers and header provided by application, and send the message to the network.

A middleware library, for instance, UA Core library, may insert its own headers in the list of named attributes.

For instance, assume that a terminal application answers to a call as follows:

```
ua_call_acccept(&call,
                SIPTAG_CALL_INFO_STR(icon_url),
                SIPTAG_CALL_INFO_STR(calling_card_url),
                TAG_END());
```

The UA core library may then add media-processing specific headers (headers related to SDP processing), headers providing information about Supported features, User-Agent, and Contact header used for the future message within the dialog. The library could implement a ua_call_reply() function as follows:

```
void ua_call_accept(ua_call_t *call,
                    tag_type_t tag, tag_value_t value,
                    ...)
{
  ta_list ta;
  ...

  ta_start(ta, tag, value);

  ...                 Do the media processing, create sdp

  nta_incoming_treply(call->irq,
                      SIP_200_OK,
                      SIPTAG_USER_AGENT_STR(call->ua->name),
                      SIPTAG_SUPPORTED(call->ua->supported),
                      SIPTAG_CONTACT(call->ua->contact),
                      SIPTAG_CONTENT_TYPE_STR("application/sdp"),
                      SIPTAG_PAYLOAD(sdp),
                      TAG_NEXT(ta_tags(ta)));

  ta_end(ta);
}
```

The NTA library may then add all the boilerplate headers to the reply, like From, To, Call-ID, CSeq, Record-Route, Timestamp, etc.

Such detail will be implementation dependent, and within the scope of one skilled in the art.

A SIP dialog is a relationship between two SIP user agents, usually but not always corresponding to an IP telephony call

between these user agents. A dialog is specified by Call-ID, local name and its tag, remote name and its tag, and by the remote contact URI. A dialog also contains a route that is used to send the requests. According to the latest SIP specification, the local and remote names are ignored (so they can be modified to better reflect the real identity of call parties after a call redirection, for instance). A dialog also specifies a route to the other SIP UA. A route is a list of SIP URI that the request must traverse, at simplest it just contains the URI of the remote UA.

In addition to the fully specified dialogs (known as legs in NTA documentation), the NTA provides dialog templates (default legs). They are underspecified, i.e., they miss some or all the attributes listed above. They can be used, for example, when a call is being set up and the remote peer is not known. Other uses include routing: an application can ask NTA to route certain request URLs to a given application object. The request URI can also contain a wildcard, for example, a gateway may want to pick up all requests to a certain domain regardless of the user part (usually containing the telephone number). A default leg can also be used when an initial route is used.

NTA can also overspecify dialogs: the application can ask that only requests with certain methods are processed by the leg object.

When a dialog is created, the application must provide NTA with enough information to specify the dialog and establish the route. While the route processing is responsibility of NTA, the application must known when the dialog is established. In basic SIP, a dialog is established with a 200-series response to INVITE, but SIP extensions like

100rel and events have their own rules for establishing a dialog.

In a preferred embodiment using signaling techniques as described hereinabove, the incoming messages are processed according to the following algorithm:

1. If the message is a request:

a) If the topmost Via header has a branch parameter starting with "z9hG4bK" (branch parameter starting with "z9hG3bK" identify a request formed according to the SIP-bis specification, and uniquely identify a transaction together with host and port value from the Via header), hash h1 is calculated from branch parameter, host and port from topmost Via header. A server transaction from hash table 1 is searched for. If a matching transaction (with the same Via branch parameter, host and port) is found, the message is given to the server transaction object to process.

If no matching transaction is found, the next step is proceeded to.

b) Hash h2 is calculated from Call-ID header and CSeq sequence number. A server transaction from hash table 2 is searched for. If a transaction (with matching Call-ID, CSeq number, RequestURI, From and To headers, and topmost Via header) is found, the message is given to the transaction object to process.

If no matching transaction is found, the next step is proceeded to.

c) If an incomplete match with different topmost Via header was found in step (b), and the transaction object has a ua or merge_482 attribute set, the message is given to the transaction object. (Request is either an ACK message

acknowledging a 200-series response to an INVITE, or an merged request).

Otherwise, the next step is proceeded to.

d) Hash h3 is calculated from the Call-ID. A matching dialog leg is searched for from hash table 3. If a leg (with matching RequestURI, Call-ID, From and To headers) is found, the request is given to the leg object to process. The leg object creates a new server transaction object (object inherits semantic attributes from leg), inserts the transaction into hash tables 1 and 2, and lets the application respond to the request.

Otherwise, the next step is proceeded to.

e) If an incomplete match is found, the request is given to the leg object to process as in step 1(d).

Otherwise, the next step is proceeded to.

Note that here, RequestURI as well as Call-ID, From or To headers may be unspecified (wildcarded).

f) Hash h4 is calculated from RequestURI. A matching default leg is searched for from hash table 4. If a leg (with matching RequestURI) is found, the request is given to the leg object to process as in the step 1(f).

Otherwise, the next step is proceeded to.

Note that here RequestURI may be unspecified (wildcarded).

g) Let the application process request statelessly. If the application does not implement stateless processing, proceed to the next step.

h) If the application does not implement stateless processing, reply with an application-provided response code (by default, 500 Internal Server Error).

2. If the message is a response:

a) It is checked that the topmost Via header is valid and has originated from this instance of NTA. If the check fails, the message is discarded.

Otherwise, the next step is proceeded to.

b) Hash h5 is calculated from branch parameter of topmost Via. A client transaction from hash table 5 is searched for. If a matching transaction (with same Via branch parameter) is found, a response message is given to the transaction object to process.

Otherwise, the next step is proceeded to.

c) The application then processes the request statelessly. If the application does not implement stateless processing, the next step is proceeded to.

d) It is checked if the response is a final response to an INVITE transaction. If it is, an ACK request is sent. If the response is a 200-series reply, a BYE is also sent immediately.

Otherwise, the message is discarded.

3. Otherwise, if the message is not a request or a response it is discarded.

Referring to Figure 5, there is shown a state diagram associated with the above described method, the operation of which will be apparent to one skilled in the art. As can be seen in Figure 5, the states shown are outgoing_recv 502, agent_recv_response 504, agent_recv_message 508, agent_recv_request 510, leg_recv 506, and incoming_recv 512.

In summary, the present invention therefore provides, in the preferred embodiments, an SIP middleware that can be used

in: SIP user agents (terminals, gateways); SIP proxies (stateful, stateless, session); SIP Back-to-back user agents; SIP registrars; SIP redirect servers (both stateful and stateless); SIP presence servers, etc.

It will be apparent from the description of the above example scenarios that various modifications to the invention are possible. The invention, and embodiments thereof, provides several advantages, some of which are stated hereafter.

Advantageously, the invention consistently uses URLs for addressing.

Embodiments of the invention use typesafe named argument lists. Named argument lists are used to transparently pass optional arguments through middleware layers (application core, transaction). Extension of functionality is possible without modifying intermediate software layers.

Attributes changing the NTA semantics means that the transaction function may be tailored for a particular application (UA/proxy).

Advantageously attributes from agent and dialog are inherited.

Embodiments of the present invention provide an extensible message factory object. The parser is provided by the application. The message syntax is extensible and can be tailored for the application. The application does not have to know about the needs of underlying layers. If the application does not use a header, it can be left unparsed (and the header parsing and manipulation code can be removed). Different applications can have different view on some headers.

Step 1 (c) in the method described above offers the advantage of the UA not needing to keep a hash table identical to h2, but the topmost Via.

Step 1 (e) in the method described above allows part of the request routing to be implemented as a part of the dialog matching code.

Step 1 (f) in the method described above allows part of the request routing to be implemented as part of the dialog matching code

Step 1 (h) in the method described above provides a failure proof fallback code for routing logic.

Step 2 (d) in the method described above provides a common fallback code for obscure failure cases.

The invention also advantageously provides the possibility to specify only minimum amounts of arguments when making a request or replying to it. Complexity is removed from application (potentially in many places) to the transaction function (implemented only once)

Whilst the present invention has been described herein by way of reference to particular examples, it is not limited to those examples. The scope of the invention is defined by the appended claims.

Claims

1. A method of providing access from an application to an underlying protocol engine at a plurality of different layers.

2. A method according to claim 1 wherein said access is provided through a transaction layer.

3. A method according to claim 2 wherein access is provided through an interface of the transaction layer at a plurality of different layers.

4. A method according to any one of claims 1 to 3 wherein access is provided at an application core layer.

5. A method according to claim 4 wherein said access is to dialogs.

6. A method according to any one of claims 1 to 5 wherein access is provided at a transaction layer.

7. A method according to claim 6 wherein access is to transactions.

8. A method according to any one of claims 1 to 5 wherein access is provided at a transport layer.

9. A method according to claim 8 wherein said access is to a stateless message transport.

10. A method according to any one of claims 1 to 9 wherein there is further additionally provided other application core functionality.

11. A method according to any one of claims 1 to 10 wherein there is further provided session initiation protocol functionality to the application.

12. A method according to any one of claims 1 to 11 wherein the application is IP telephony.

13. A method according to claim 11 or claim 12 in which a request message is processed in dependence on a comparison of the top most Via header of the message.

14. A method according to any one of claims 1 to 13 in which for a request message at least part of the request routing is implemented as part of a dialog matching code.

15. A method according to any one of claims 11 to 14 in which for a response or request message there is provided a failure-proof fallback code for routing logic.

16. A method according to any one of claims 1 to 15 in which the application is a multimedia application.

17. An interface adapted to provide access between an application and an underlying protocol engine at a plurality of different layers.

18. An interface according to claim 17,wherein the interface is formed in transaction layer.

19. An interface according to claim 18, wherein the transaction layer is adapted to provide an interface at the plurality of different layers.

20. An interface according to claim 19 wherein an interface is provided at an application core layer.

21. An interface according to claim 19 or 20 wherein the interface is provided at a transaction layer.

22. An interface according to any one of claims 18 to 20 wherein the interface is provided at a transport layer.

23. An interface according to any one of claims 17 to 22 wherein there is further provided means for implementing further application core functionality above the interface.

24. An interface according to any one of claims 17 to 23 wherein there is further provided means below the interface functionality.

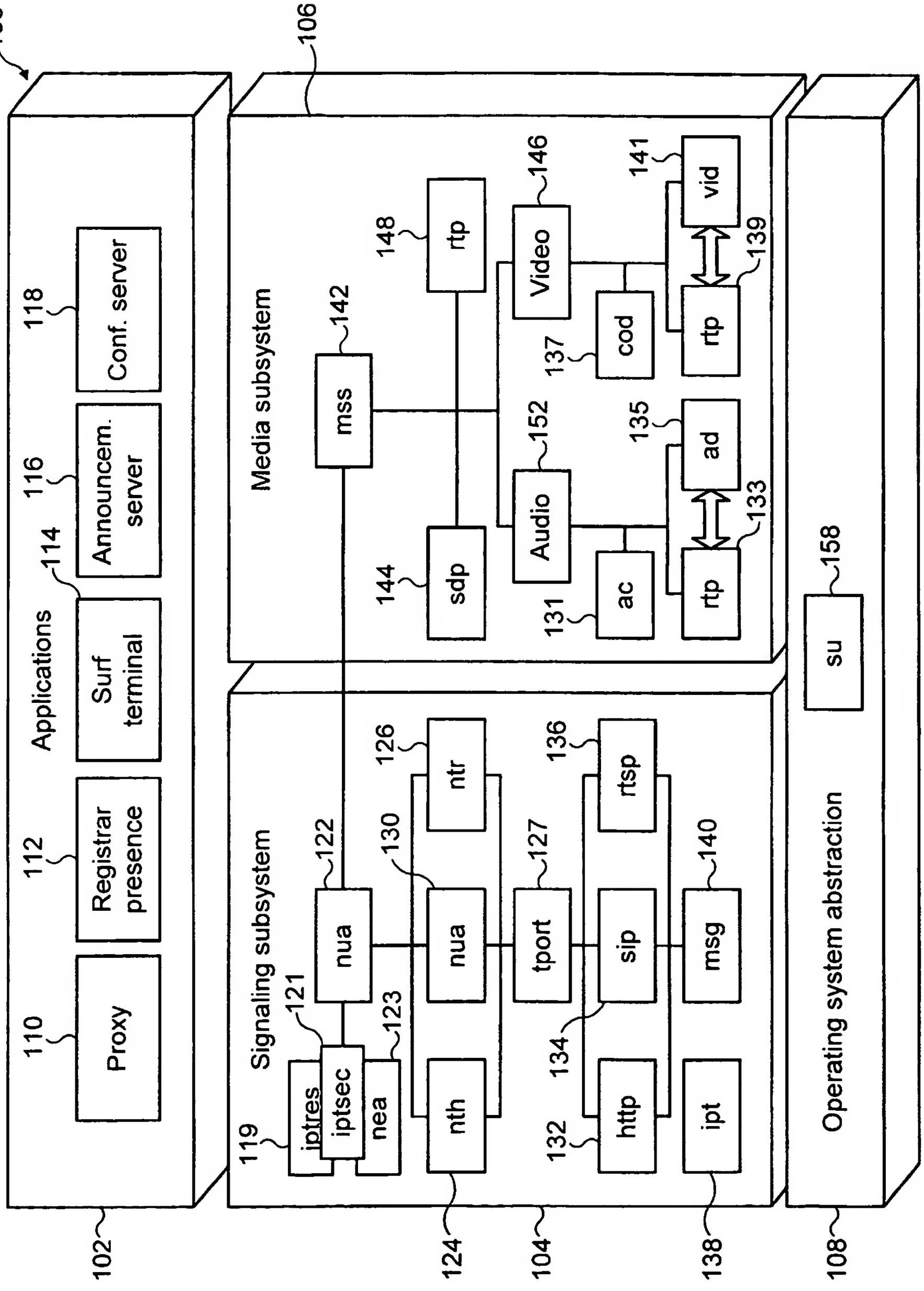

100
106
Applications
118
Conf. server
116
Announcem. server
114
Surf terminal
112
Registrar presence
110
Proxy
102
Media subsystem
148
rtp
146
Video
141
vid
139
142
mss
137
cod
rtp
152
Audio
135
ad
144
sdp
131
ac
rtp
133
158
su
Signaling subsystem
126
ntr
136
rtsp
122
nua
130
nua
127
tport
sip
140
msg
121
iptres
iptsec
123
nea
119
nth
124
http
134
132
ipt
138
104
Operating system abstraction
108

FIG. 1

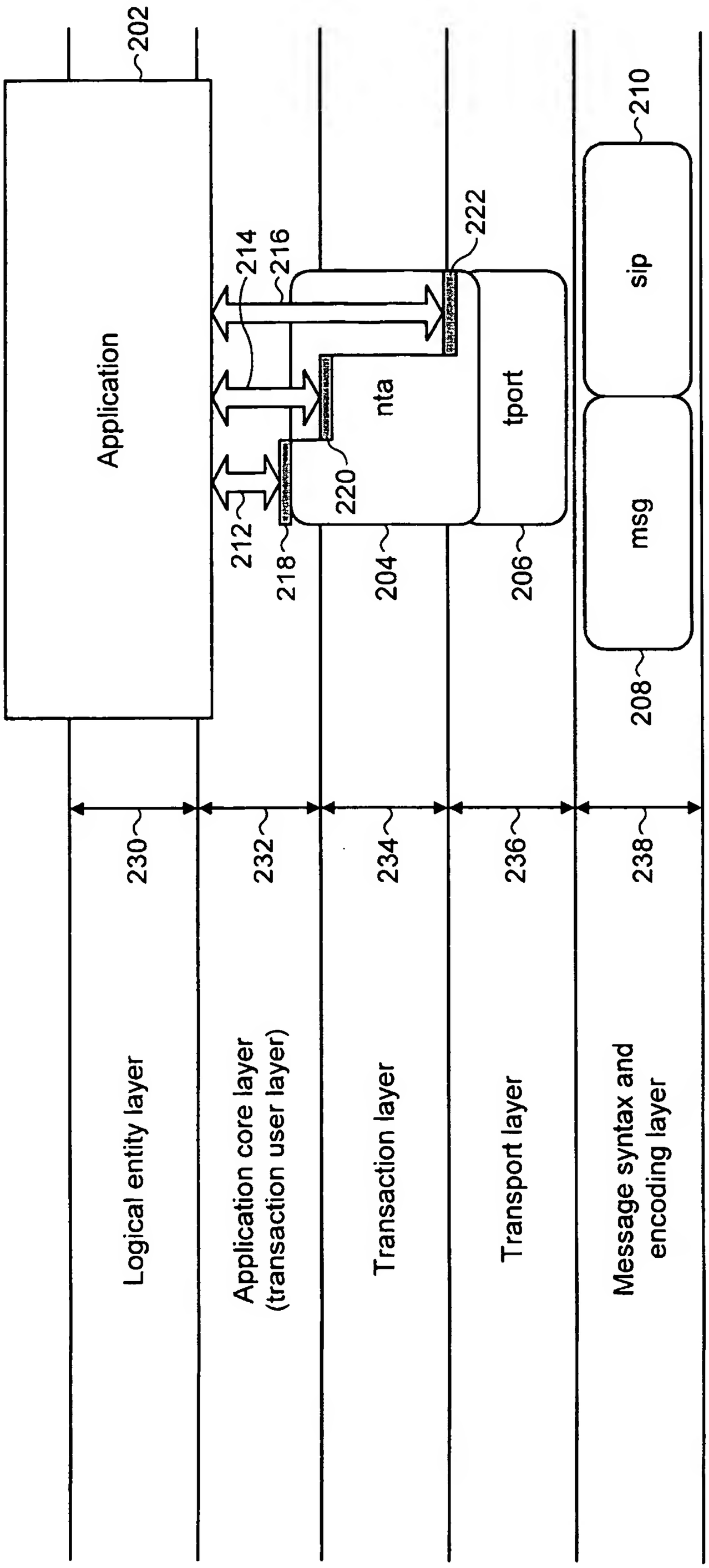

202
Application
214
216
222
nta
tport
sip
210
212
220
218
204
206
msg
208
230
232
234
236
238
Logical entity layer
Application core layer
(transaction user layer)
Transaction layer
Transport layer
Message syntax and
encoding layer

FIG. 2

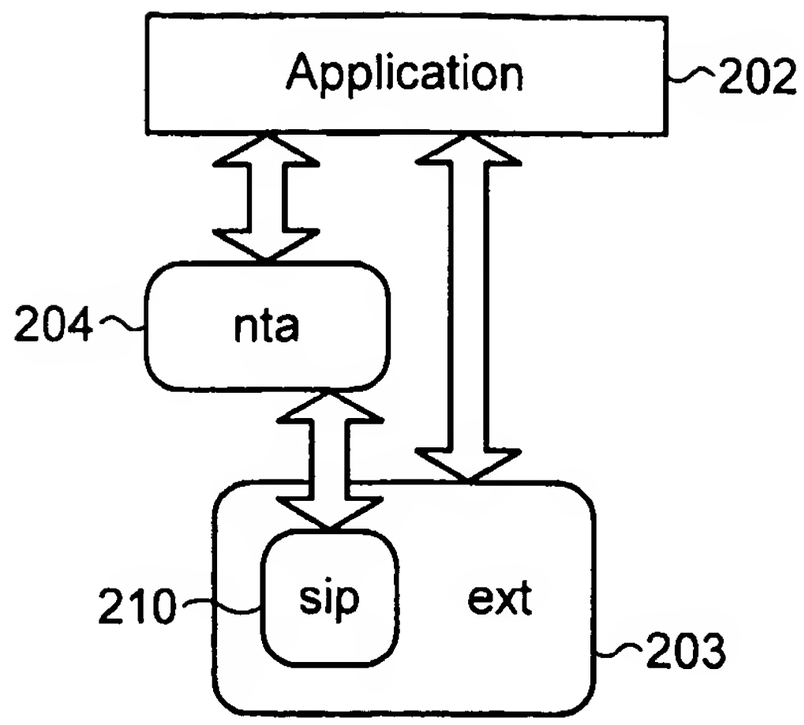

Application
202
204
nta
210
sip
ext
203

FIG. 3

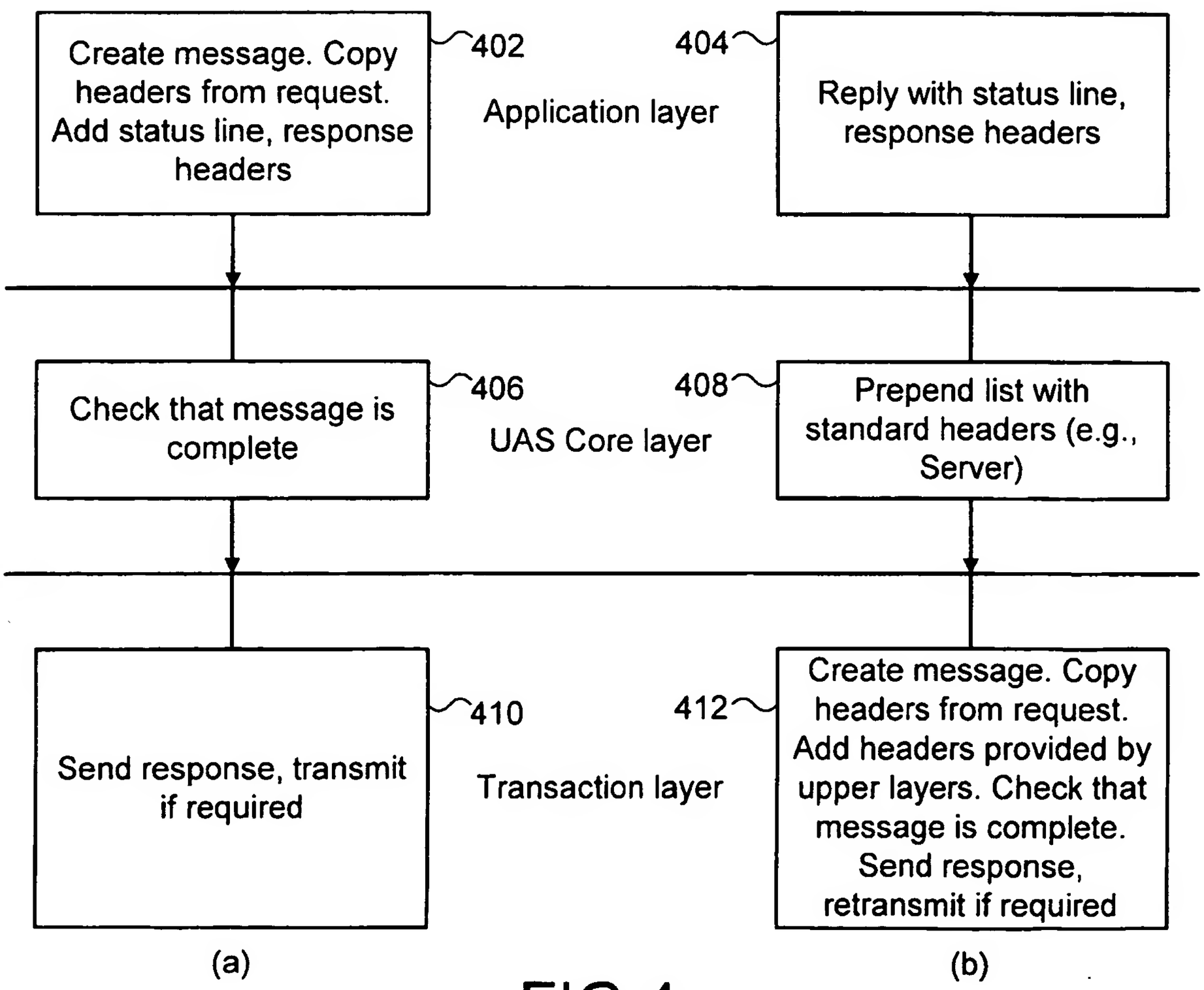

Create message. Copy headers from request. Add status line, response headers
402
Application layer
404
Reply with status line, response headers
Check that message is complete
406
UAS Core layer
408
Prepend list with standard headers (e.g., Server)
Send response, transmit if required
410
Transaction layer
412
Create message. Copy headers from request. Add headers provided by upper layers. Check that message is complete. Send response, retransmit if required
(a)
(b)

FIG.4

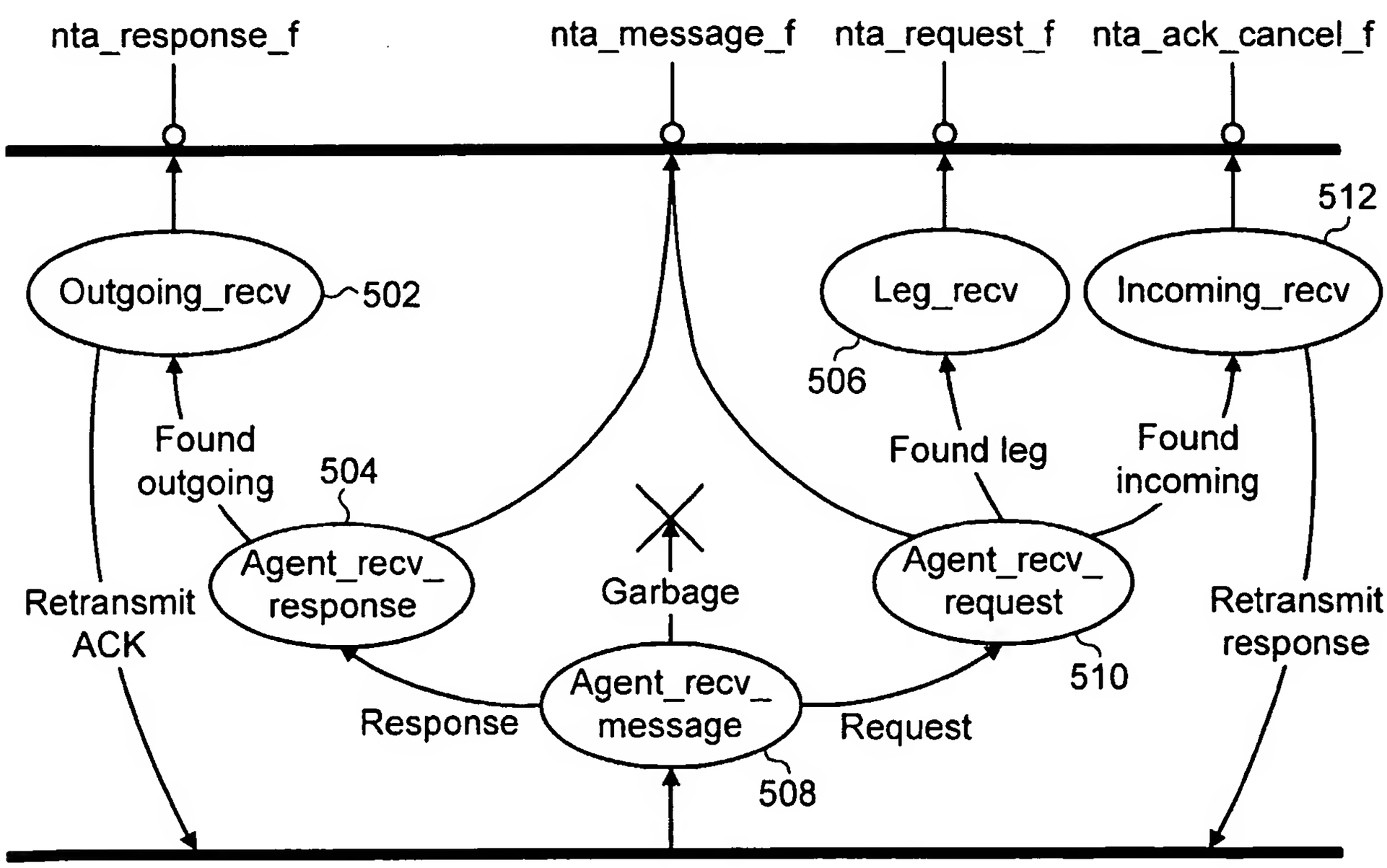

nta_response_f
nta_message_f
nta_request_f
nta_ack_cancel_f
Outgoing_recv
502
Leg_recv
506
Incoming_recv
512
Found outgoing
504
Found leg
Found incoming
Agent_recv_ response
Garbage
Agent_recv_ request
510
Retransmit ACK
Retransmit response
Agent_recv_ message
Response
Request
508

FIG. 5

INTERNATIONAL SEARCH REPORT

PCT/IB 02/02936

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F17/00 H04L29/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
INSPEC, EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	HUA ZOU ET AL: "Prototyping SIP-based VoIP services in Java" WCC 2000 - ICCT 2000. 2000 INTERNATIONAL CONFERENCE ON COMMUNICATION TECHNOLOGY PROCEEDINGS (CAT. NO.00EX420), PROCEEDINGS OF 16TH INTERNATIONAL CONFERENCE ON COMMUNICATION TECHNOLOGY (ICCT'00), BEIJING, CHINA, 21-25 AUG. 2000, pages 1395-1399 vol.2, XP002227133 2000, Piscataway, NJ, USA, IEEE, USA ISBN: 0-7803-6394-9 page 1, column 2, line 32-41 page 2, column 1, line 17-38 page 2, column 2, line 40 -page 3, column 2, line 39 abstract; figures 1,2 --- -/--	3-9, 11-16, 18-24

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
14 January 2003	06 02 2003

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	ISMAR HADZIEFENDIC/JAA

INTERNATIONAL SEARCH REPORT

PCT/IB 02/02936

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JAIN R ET AL: "Java/sup TM/ call control (JCC) and session initiation protocol (SIP)" IEICE TRANSACTIONS ON COMMUNICATIONS, DEC. 2001, INST. ELECTRON. INF. & COMMUN. ENG, JAPAN, vol. E84-B, no. 12, pages 3096-3103, XP002227134 ISSN: 0916-8516 page 1, column 1, line 45-54 page 1, column 2, line 31-39 page 2, column 1, line 19-57 page 2, column 2, line 12-16 page 3, column 2, line 21-38 abstract; figures 3-6 ---	3-9, 11-16, 18-24
A	HIROSHI SASAKI-NEC: "Java TM Call Controll v1.0 to Session Initiation Protocol Mapping" JAIN TM COMMUNITY, [Online] 25 October 2001 (2001-10-25), XP002227135 Retrieved from the Internet: <URL:http://java.sun.com/products/jain/JCC2SIP.pdf> [retrieved on 2003-01-14] page 5 -page 6 ---	3-9, 11-16, 18-24
A	EP 0 969 628 A (TOKYO SHIBAURA ELECTRIC CO) 5 January 2000 (2000-01-05) page 2, column 1, line 1 -page 7, column 12, line 44 abstract; figure 2 ---	3-9, 11-16, 18-24
A	US 2001/055315 A1 (HU QI) 27 December 2001 (2001-12-27) page 2, column 2, line 38 -page 3, column 1, line 15 abstract; figures 1-3,6 -----	3-9, 11-16, 18-24

INTERNATIONAL SEARCH REPORT

PCT/IB 02/02936

Box I Observations where certain claims were found unsearchable (Continuation of item 1 of first sheet)

This International Search Report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☒ Claims Nos.: 1,2,10,17
because they relate to parts of the International Application that do not comply with the prescribed requirements to such an extent that no meaningful International Search can be carried out, specifically:
see FURTHER INFORMATION sheet PCT/ISA/210

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box II Observations where unity of invention is lacking (Continuation of item 2 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

1. ☐ As all required additional search fees were timely paid by the applicant, this International Search Report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this International Search Report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this International Search Report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest.

☐ No protest accompanied the payment of additional search fees.

FURTHER INFORMATION CONTINUED FROM PCT/ISA/ 210

Continuation of Box I.2

Claims Nos.: 1,2,10,17

The scope of protection for the subject mattter as defined by the claims 1-2,10 and 17 is unclear.

The applicant's attention is drawn to the fact that claims, or parts of claims, relating to inventions in respect of which no international search report has been established need not be the subject of an international preliminary examination (Rule 66.1(e) PCT). The applicant is advised that the EPO policy when acting as an International Preliminary Examining Authority is normally not to carry out a preliminary examination on matter which has not been searched. This is the case irrespective of whether or not the claims are amended following receipt of the search report or during any Chapter II procedure.

INTERNATIONAL SEARCH REPORT

PCT/IB 02/02936

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0969628 A	05-01-2000	JP 2000156683 A	06-06-2000
		EP 0969628 A2	05-01-2000
US 2001055315 A1	27-12-2001	NONE	